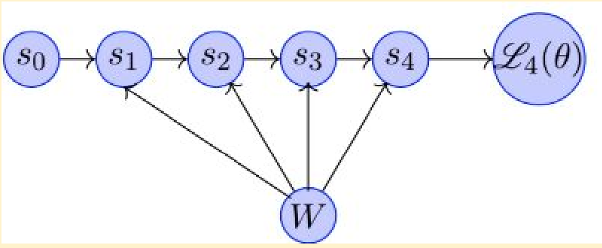
PadhAI: Long Short Term Memory (LSTM) and Gated Recurrent Units (GRUs).

One Fourth Labs

Learnings so far:

When we have a recurrent neural network, as shown below

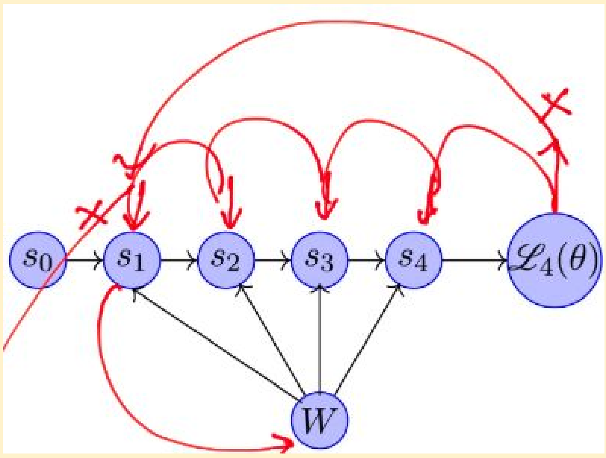


Suppose we calculate loss at 5th time step

The loss was generated because the s_1 was not computed properly, hence s_2 was not computed properly, in the same way the value of ***states*** till time **step 5** were all not computed properly.

s_1 was not computed properly because **W** on which s_1 depends was not in a right configuration**.**

The loss at time step 5 is high because the **weights in weight matrix(W)** are not in a correct configuration, this feedback needs to go back to **W through** s_1 **,** s_1 is **not good** hence **W** is **not good enough, so W needs to change.**



To give the feedback, feedback needs to travel through s_4 all the way up to s_1 and then finally to **W.**

In this we saw that, if we have such a long chain, we can have 2 types of problems

1. **Vanishing gradients:**
   Gradient is going to be a product of many terms and if all these terms are **very small** then the **gradients will vanish.**

2. **Exploding Gradients:**
   Gradient is going to be a product of many terms and if all these terms are **very large** then the **gradients will explode.**

Problems with RNNs.

- At each new timestep the old information gets morphed by the current input
- One could imagine that after t steps the information stored at time step t − k (for some k < t) gets completely morphed

- Even during backpropagation the information does not flow well

White Board Analogy.

Have a look at the image of the **white-board** below.

As u can see that when we keep on putting **new information** at **every new time step** on to the **board**, it becomes **really hard** to **extract or determine** the information that was put on the board at the first-time step.

Problem:

In the very same way, when we have long sequences and we want the initial input (say x_1) to contribute to the output, ***it becomes difficult to find out how is*** x_1 ***contributing to the output.***

Solution to the problem:

Strategy

- Selectively write on the board
- Selectively read the already written content
- Selectively forget (erase) some content

Let's try to understand our strategy using the following example.

We have the values of **a, b, c and d** as follows

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

To compute the above equation, we need to follow the series of calculations given below.

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

We calculate first three steps, and write them on our white board

$$\begin{aligned} ac &= 5 \\ bd &= 33 \\ bd + a &= 34 \end{aligned}$$

After calculating the value of **bd**, we have used it in the equation **bd + a,** now we do not require to ***store or remember*** the value of **bd** any longer.

Here's where ***selective forget*** comes, we erase the content which is no longer useful (value of **bd**), and replace it with new information/content.

$$ac = 5$$
$$ac(bd + a) = 170$$
$$bd + a = 34$$

Notice value of bd is selectively forgotten and replace with information of an another equation.

Now, from the previously calculated results we will calculate

$$ac(bd + a) = 170$$

To calculate $ac(bd + a)$ we had to ***selectively read the already written content***, on the white board, the values of **ac** and **bd + a** only.

Now we will calculate the value of **ad**, we can erase any of the two values of **ac** and (**bd + a**) using ***selective forget***, as we no longer need them, and replace it with the value of **ad.**

$$ac = 5$$
$$ac(bd + a) = 170$$
$$ad = 11$$

Observe the use of ***selectively write strategy***.

As you can see in all the above steps, we are **selectively writing** the overall result of all the equations instead of breaking and writing the results of even the sub parts of the equations. ***Selectively write strategy*** helps us to express main fundamentals in a very concise way using very less space.

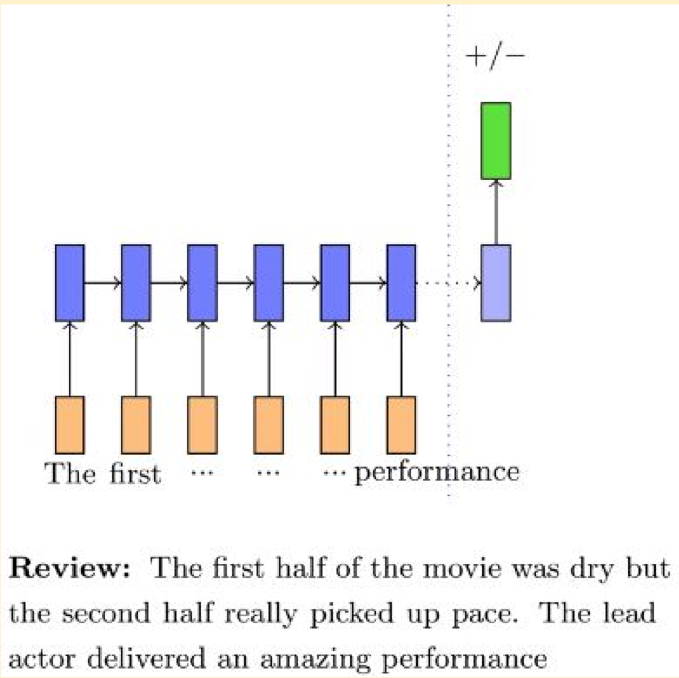
Now, we calculate the result of final equation, keeping all the necessary content on the board and removing all the unnecessary content.

$$ad + ac(bd + a) = 181$$
$$ac(bd + a) = 170$$
$$ad = 11$$

To calculate **ad + ac(bd + a)** we needed the values of **ac(bd + a)** and **ad,** so we kept them on the board and deleted all the other unnecessary values.

Our strategy helps us to keep all the necessary values in use and discard unnecessary values which are no longer required, keeping the process of reaching the final goal much efficient and tidy.

Real world example of longer sequences.



The review statement initially gives a negative impression about the movie, but somewhere in the middle, the review statement tells us how movie changed its pace and also tells us about the great performance given by the lead actor.

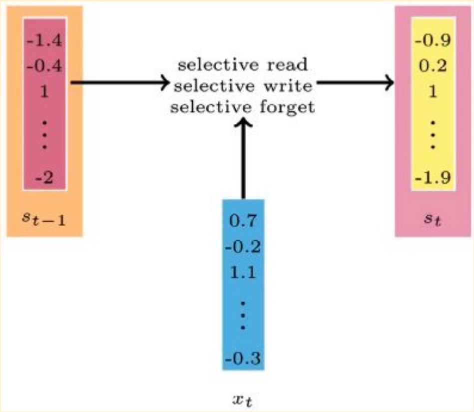
Overall, we want to classify this review as a positive review.

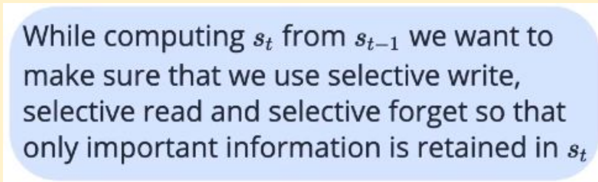
Ideally, we want to

- forget the information added by stop words (a, the, etc.)
- selectively read the information added by previous sentiment bearing words (awesome, amazing, etc.)
- selectively write new information from the current word to the state

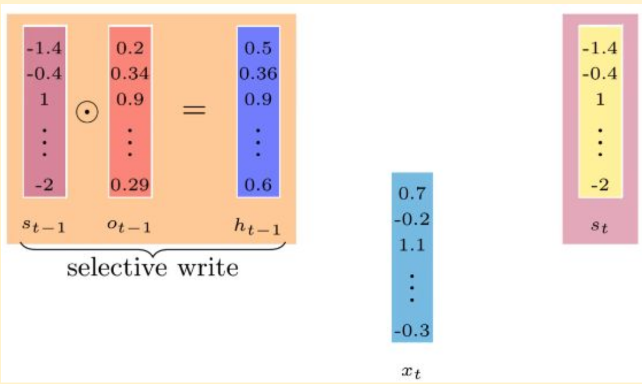
Going back to RNNs

LSTM cells





<u>Selective Write in RNNs.</u>



Note: In the above diagram the values of h_{t-1} are just for example, they are not the correct answers of $s_{t-1} \odot o_{t-1}$.

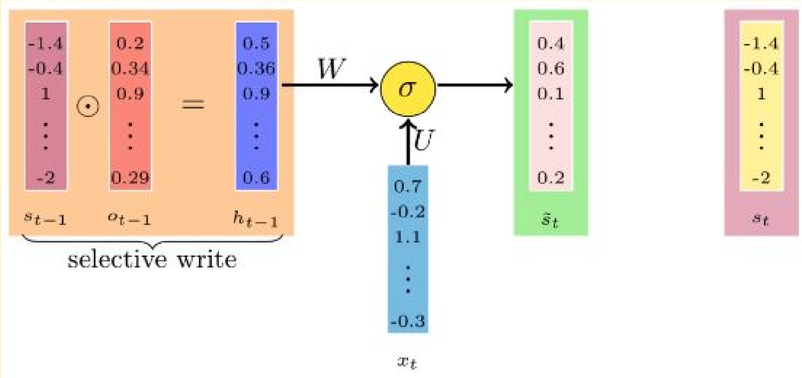
o_{t-1} **tells what proportion of the current state to pass on to the next state.**

But how do we compute o_{t-1} and How does RNN know what fraction of the state to pass on?

- learn o_{t-1} from data
- the only thing that we learn from data is parameters
- **Solution:** express o_{t-1} using parameters

$$o_{t-1} = \sigma(U_o x_{t-1} + W_o h_{t-2} + b_o)$$
$$h_{t-1} = s_{t-1} \odot o_{t-1}$$

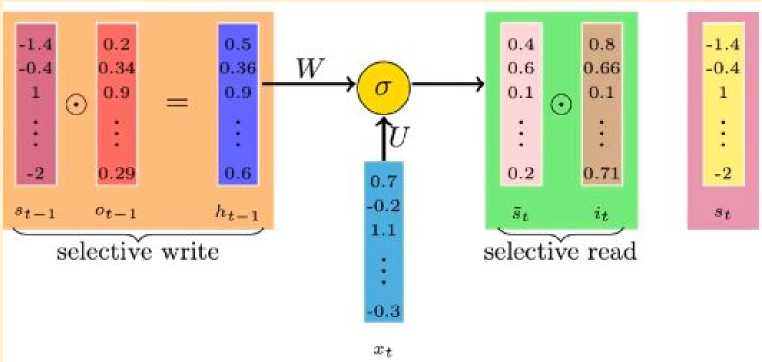
o_t is called the **output** gate



$$\tilde{s}_t = \sigma(Ux_t + Wh_{t-1} + b)$$

- $\tilde{s}_t$ thus captures all the information from the previous state h_{t-1} and the current input x_t

Introducing Selective Read



$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i)$$
$$= \tilde{s}_t \odot i_t$$

i_t is called the **input** gate

Summary so far

Previous state:

s_{t-1}

Output gate:

$o_{t-1} = \sigma(W_o h_{t-2} + U_o x_{t-1} + b_o)$

Selectively Write:

$h_{t-1} = o_{t-1} \odot \sigma(s_{t-1})$

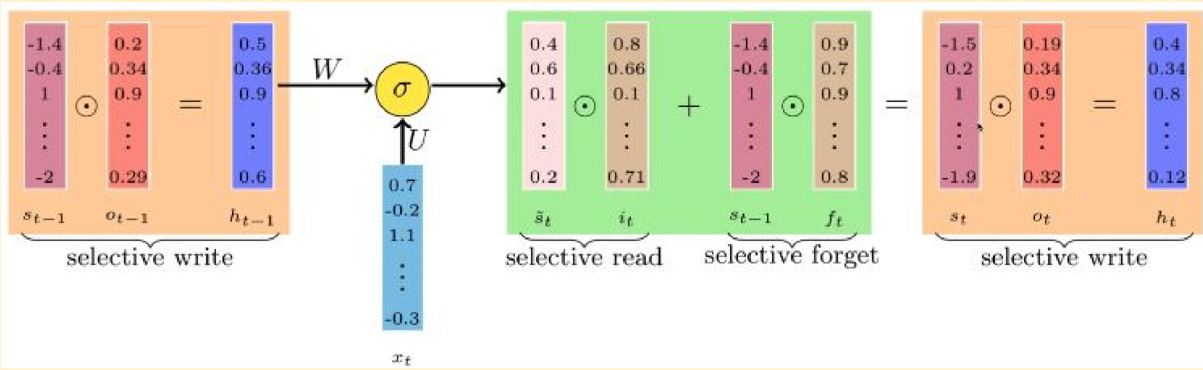
Current (temporary) state:

$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$

Input gate:

$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$

Selective Forget



$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$
$$s_t = \tilde{s}_t \odot i_t + s_{t-1} \odot f_t$$

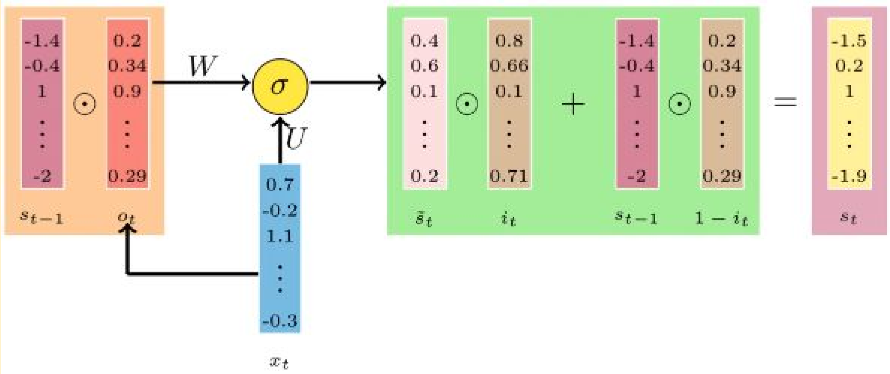
Where f_t is called forget gate.

Full set of Equations

Gates:	**States:**
$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$	$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$
$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$	$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$
$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$	$h_t = o_t \odot \sigma(s_t)$

Gated Recurrent Units

- LSTM has many variants which include different number of gates and also different arrangement of gates
- The one which we just saw is one of the most popular variants of LSTM
- Another equally popular variant of LSTM is Gated Recurrent Unit which we will see next



Gates:

$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$

$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$

States:

$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$$

$$s_t = (1 - i_t) \odot s_{t-1} + i_t \odot \tilde{s}_t$$